Oblig 1

Greger Svenn

October 1, 2015

Ex. 1

The momentum equation and the incompressible continuity are given by:

$$\frac{d\vec{u}}{dt} + (\nabla(\vec{u}\vec{u}) = \nabla \cdot (\nu\nabla\vec{u}) - \nabla p \tag{1}$$

$$\nabla\vec{u} = 0 \tag{2}$$

Rearranging (1) to

$$\frac{d\vec{u}}{dt} + \nabla(\vec{u}\vec{u}) - \nabla \cdot (\nu\nabla\vec{u}) = -\nabla p \tag{3}$$

, our left side looks similar to the PISO solvers momentum predictor

```
// Momentum predictor

            fvVectorMatrix UEqn
            (
                fvm::ddt(U)
              + fvm::div(phi, U)
              + turbulence->divDevReff(U)
            );
```

We introduce guessed values for u and p called $u^\star$ and $p^\star$. Then set $u^{k-1} = u^\star$.
To handle the non linear convection term we use an iterative solution and say that

$$\nabla(\vec{u}\vec{u}) \approx \nabla(\vec{u}^{k-1}\vec{u}^\star) \tag{4}$$

```
              + fvm::div(phi, U)
```

Here phi $= \vec{u}^{k-1}$ and $\vec{u}^\star$ is the new u that we want to solve. The convection term is now reduced to one known and one unknown, thus the term is linear. (3) is solved in pisoFoam by using the pressure from previous time step as, follows:

```
            if (momentumPredictor)
            {
                solve(UEqn == -fvc::grad(p));
            }
```

By discretizing the momentum equation and substituting u in to the continuity equation, we can derive an equation for pressure.

$$a_p\vec{u}_p + \sum_{nb} a_{nb}\vec{u}_{nb} = \vec{r} - \nabla p \tag{5}$$

$$\vec{H}(\vec{u}) = \vec{r} - \sum_{nb} a_{nb} \tag{6}$$

$$\vec{u}_p = (a_p)^{-1}(\vec{H}(\vec{u}) - \nabla p) \tag{7}$$

$$\nabla \cdot [(a_p)^{-1}\vec{H}(\vec{u})] = \nabla \cdot [(a_p)^{-1}\nabla p] \tag{8}$$

```
volScalarField rAU(1.0/UEqn.A());

volVectorField HbyA("HbyA", U);
HbyA = rAU*UEqn.H();
surfaceScalarField phiHbyA
(
    "phiHbyA",
    (fvc::interpolate(HbyA) & mesh.Sf())
  + fvc::interpolate(rAU)*fvc::ddtCorr(U, phi)
);

adjustPhi(phiHbyA, U, p);
```

In pisoFoam the a_p coefficient is calculated (first line), then the solver stores $(a_p)^{-1}\vec{H}(\vec{u})$ in a field HbyA.

Calculate flux:

```
(
    "phiHbyA",
    (fvc::interpolate(HbyA) & mesh.Sf())
  + fvc::interpolate(rAU)*fvc::ddtCorr(U, phi)
);

adjustPhi(phiHbyA, U, p);
```

Then calculate a new pressure for the prescribed number of non-orthogonal corrector steps

```
for (int nonOrth=0; nonOrth<=nNonOrthCorr; nonOrth++)
{
    // Pressure corrector

    fvScalarMatrix pEqn
    (
        fvm::laplacian(rAU, p) == fvc::div(phiHbyA)
    );

    pEqn.setReference(pRefCell, pRefValue);
```

Then if phi/u^{k-1} = U/$u^{\star}$ solver is done, else it updates u^{k-1} to the most recent u

```
    {
        phi = phiHbyA - pEqn.flux();
    }
```

Ex. 2

The simulation uses the LES model `oneEqEddy` to simulate the behavior of k. From the file

`$FOAM_SRC/turbulenceModels/incompressible/LES/oneEqEddy/oneEqEddy.H`

we get the following description:

```
Description
    One Equation Eddy Viscosity Model for incompressible flows

    Eddy viscosity SGS model using a modeled balance equation to simulate the
    behaviour of k, hence,
    \verbatim
        d/dt(k) + div(U*k) - div(nuEff*grad(k))
        =
        -D:B - ce*k^(3/2)/delta

    and

        B = 2/3*k*I - 2*nuSgs*dev(D)
        Beff = 2/3*k*I - 2*nuEff*dev(D)

    where

        D = symm(grad(U));
        nuSgs = ck*sqrt(k)*delta
        nuEff = nuSgs + nu
    \endverbatim

SourceFiles
    oneEqEddy.C

\*---------------------------------------------------------------------------*/
```

`d/dt(k)` ddescribes the change of the turbulent kinetic energy, `div(U*k)` is convection and `div(nuEff*grad(k))` is diffusion. The `-D:B - ce*k^ (3/2)/delta` are source terms describing the decay of turbulence.

All simulations uses parameters; endtime = 1 and deltaT =1e-5
Boundary conditions used for velocity

- At the walls the there is no velocities in any directions, the No-slip condition:

```
    walls
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
```

- Inlet, we've got the inlet velocity in x-direction set to 10 m/s

```
        inlet
    {
        type            fixedValue;
        value           uniform (10 0 0);
    }
```

- Outlet. Neumann condition. No velocity change at the outlet boundary

```
        outlet
    {
        type            zeroGradient;
    }
```

The pressure boundaries:

- Inlet, Neumann condition, no change in pressure at the boundary face

```
    inlet
    {
        type            zeroGradient;
    }
```

- Outlet pressure is zero, from exercise text

```
    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }
```

- At the walls the velocity No-Slip condition is inserted into the x-component of the momentum equation, combined with the continuity eqn. . This gives us $-\frac{dp}{dy} = 0$, which can be used to prove that the pressure gradient is zero.

```
    walls
    {
        type            zeroGradient;
    }
```

Mean velocity at different grid size and flow duration of 1 second

Figure 1: Rough grid, (100 10 1)

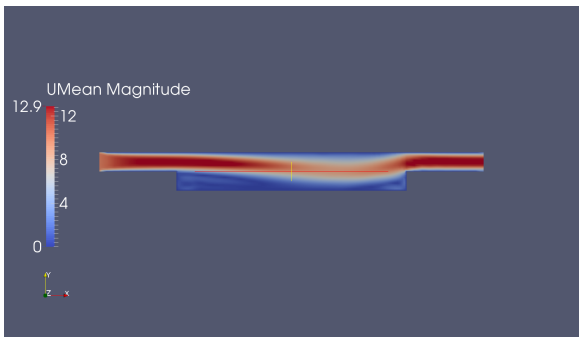


Figure 2: Less rough grid, (150 15 1)

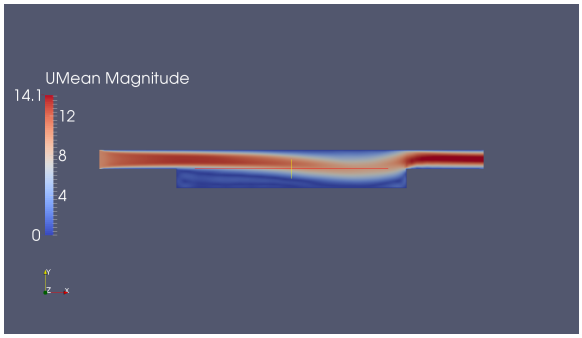
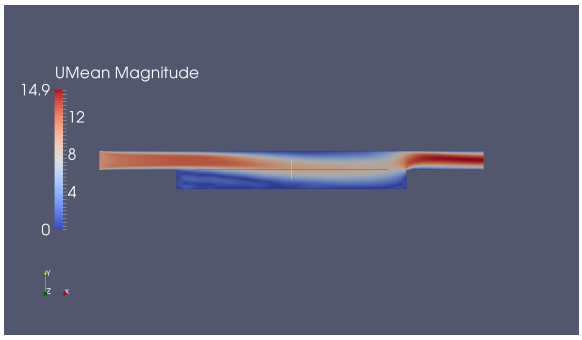


Figure 3: Fine grid, (200 20 1)



Mean pressure at different grid size and flow duration of 1 second

Figure 4: Rough grid, (100 10 1)

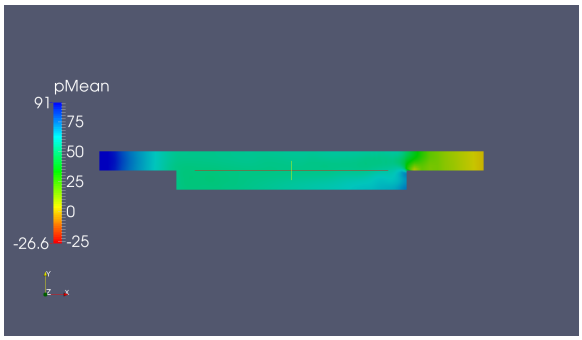


Figure 5: Less rough grid, (150 15 1)

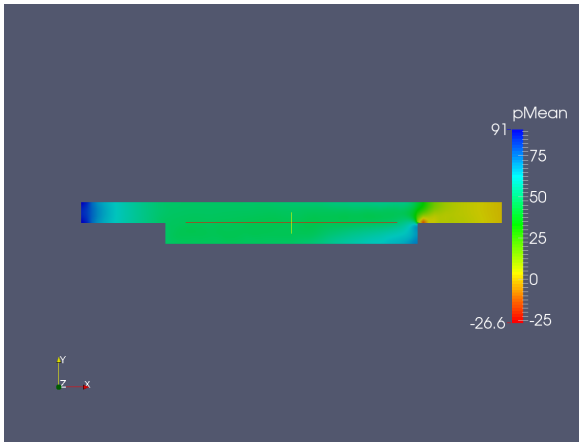
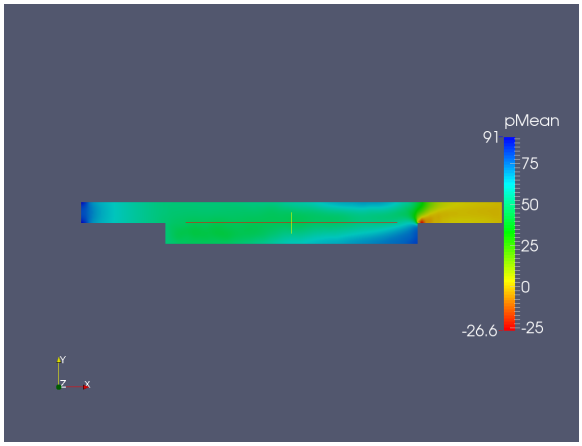


Figure 6: Fine grid, (200 20 1)



For my convection TVD scheme I had to choose a scheme that is stable for my case. The criteria for stability are different for the different schemes.

$$r = \frac{\phi_p - \phi_W}{\phi_E - \phi_P} \tag{9}$$

My assumption is that since we are using a fine mesh, the difference between the cell nodes are small, giving $r \approx 1$. For my convection TVD scheme I've therefore used `Gauss Linear`, which criteria for stability is $r < 2$.

From what we can see by comparing the mean-figures above, there is a small variation, noticeable if you look close. It would possibly be more noticeable if the end time were to be decreased. So the mean U and p are quite stable for different grids.
Comparing the actual velocity simulation, the different grids differs way more.

The solution is dependent on mesh size and Δt. To get a decent simulation you need to have a fine mesh.

Ex. 3

(second RANS solver I was planing on using was $k - \sigma$, but I haven't gotten to it.)

I've used the $k - \epsilon$ RANS model which gives a general description of the turbulent flow. Making the assumption that the ratio between the Reynolds stress an rate of deformation is the same in all directions, meaning the turbulent viscosity is isotropic.
It uses the change in kinetic energy to simulate the characteristics of turbulent flow.

The simulation uses two equations, one to determine the energy in the turbulence (k), and the other determines the dissipation of the rate of turbulent kinetic energy.

The first (k):

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial(\rho k u_i)}{\partial x_i} = \frac{\partial}{\partial x_j}[\frac{\mu_t}{\sigma_k}\frac{\partial k}{\partial x_j}] + 2\mu_t E_{ij}E_{ij} - \rho\epsilon \tag{10}$$

The sceond:

$$\frac{\partial(\rho \epsilon)}{\partial t} + \frac{\partial(\rho \epsilon u_i)}{\partial x_i} = \frac{\partial}{\partial x_j}[\frac{\mu_t}{\sigma_\epsilon}\frac{\partial \epsilon}{\partial x_j}] + C_1 \, _\epsilon \frac{\epsilon}{k} 2\mu_t E_{ij}E_{ij} - C_{2\epsilon}\rho\frac{\epsilon^2}{k} \tag{11}$$

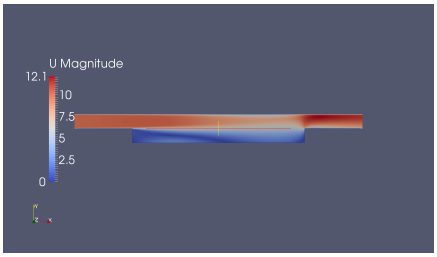
Comparing these equations to the `kEpsilon` solver description that follows theC_μ in $\mu_t = \rho C_\mu \frac{k^2}{\epsilon}$(eddy viscosity) equals `Cmu`, `C1` $= C_{1\epsilon}$, `C2` $= C_{2\epsilon}$

and `sigmaEps` = σ_ϵ. These values have been arrived from numerous iterations. ($\sigma_k = 1$ and is not used in the solver) E_{ij} represents the rate of deformation.

```
Description
    Standard k-epsilon turbulence model for incompressible flows.

    Reference:
    \verbatim
        Launder, B.E., and Spalding, D.B.,
        "Mathematical Models of Turbulence",
        Academic Press, 1972.

        Launder, B.E., and Spalding, D.B.,
        "Computational Methods for Turbulent Flows",
        Comp. Meth in Appl Mech & Eng'g, Vol 3, 1974, pp 269-289.
    \endverbatim

    The default model coefficients correspond to the following:
    \verbatim
        kEpsilonCoeffs
        {
            Cmu         0.09;
            C1          1.44;
            C2          1.92;
            sigmaEps    1.3;
        }
    \endverbatim
```

U and p boundary is the same as i Ex. 2.

(a) Velocity, fine grid, (200 20 1)



(b) Velocity, rough grid, (100 10 1)

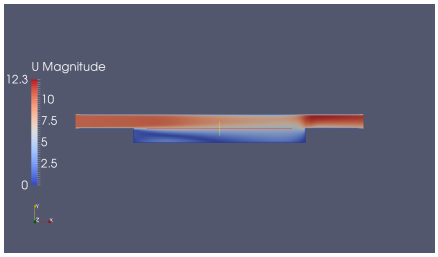


Figure 8: kinetic energy, rough grid, (200 20 1)

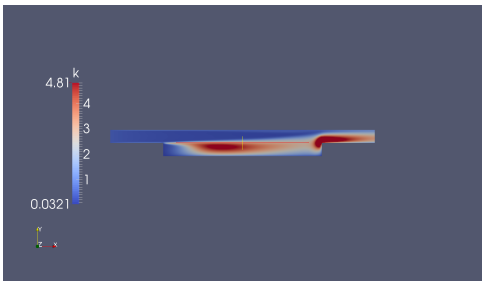
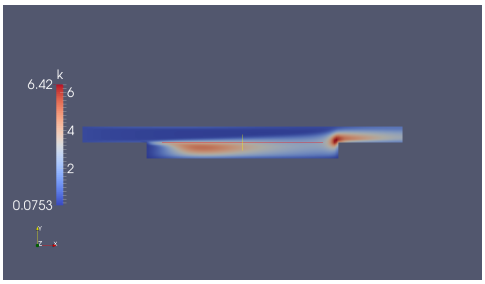


Figure 9: kinetic energy, rough grid, (100 10 1)



Bounded gauss upwind as convection scheme.

Comparing the grids roughness, there is very little difference between the two. By first look there seems to be some change in the kinetic energy, but by looking at the colour mapping, they seem to be similar.

The mesh solution is independent, because it's not time dependent (Steady state). When it has converged it will not change noticeably, therefore it stops.

Ex. 4

pimpleFoam uses large time steps, while LES is most accurate at small time steps. Thus pimpleFoam is more suited for RANS (time averaging).

Ex. 5

RANS is based on time averaging, and from what it looks like from the U solver in simple, SIMPLE is a steady state N-S solver.
Therefore it would be reasonable to compare the averaged velocities from the LES solver with the SIMPLE case when it has converged.

Figure 10: RANS/Simple

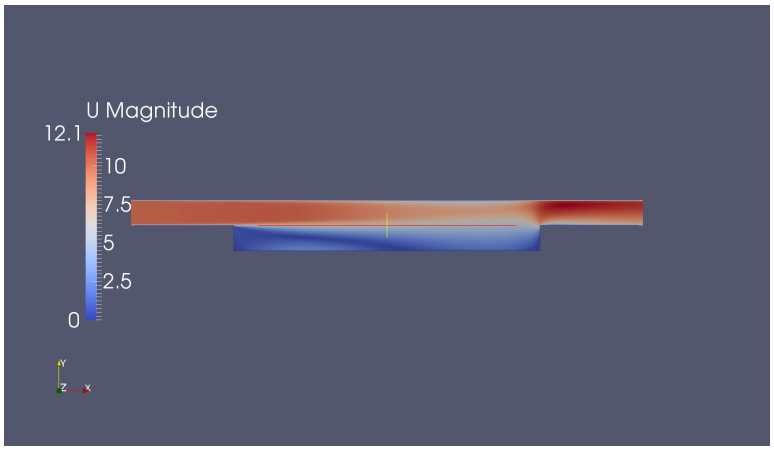
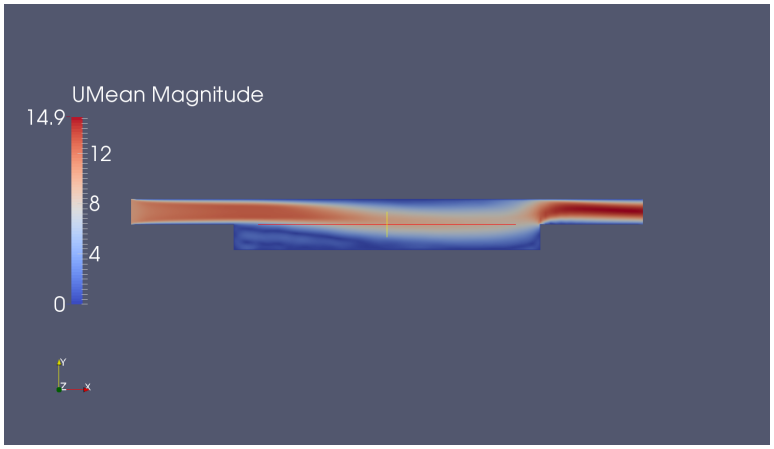


Figure 11: U mean LES model



There are some similarities between the two models, looking at the the converged RANS and mean velocities from LES

Figure 12: Recirculation bubble, RANS/Simple

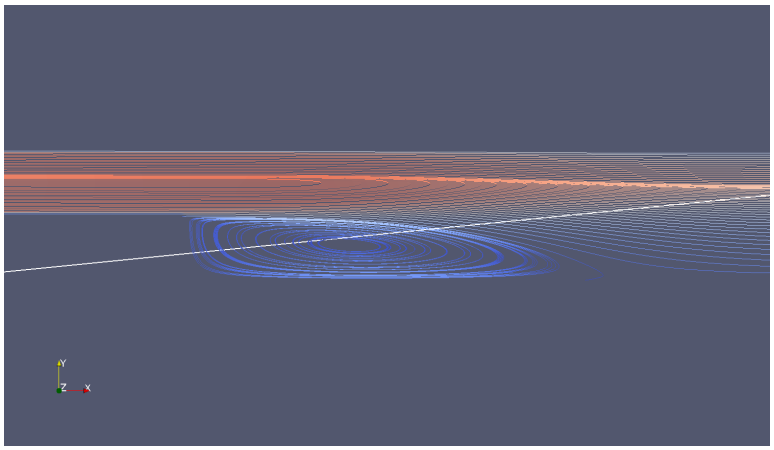
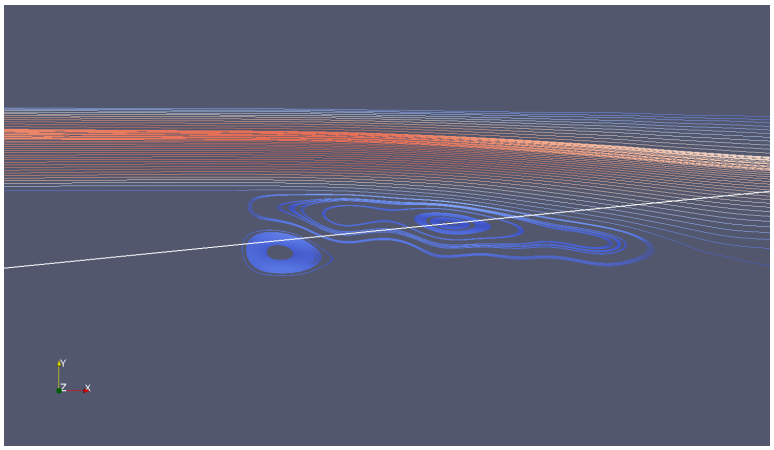


Figure 13: Recirculation bubble, LES



Comparing the two recirculation bubbles, and noticing what is said in the next exercise, we can see some similarities and understand the part about 'mimics the macroscopic effects'.

Ex. 6

Since LES is a model for turbulence and turbulence is a three dimensional phenomenon, the result from a 2D LES simulation would not be sensible. The RANS model implies that the simulation cannot represent the turbulence represent turbulence in all directions. It mimics its macroscopic effects. And the fact that, at least the $k - \epsilon$ model, uses the assumption that the ratio between the Reynolds stress an rate of deformation is the same in all

directions, there is no difference if we use 3D or 2D, it is still the same in the necessary directions.